J Real-Time Image Proc (2017) 13:581–589
DOI 10.1007/s11554-016-0612-0

SPECIAL ISSUE PAPER

Perceptually inspired real-time artistic style transfer for video stream

Dongwann Kang[1] · Feng Tian[1] · Sanghyun Seo[2]

Received: 1 January 2016 / Accepted: 6 June 2016 / Published online: 27 June 2016

Abstract This study presents a real-time texture transfer method for artistic style transfer for video stream. We propose a parallel framework using a T-shaped kernel to enhance the computational performance. With regard to accelerated motion estimation, which is necessarily required for maintaining temporal coherence, we present a method using a downscaled motion field to successfully achieve high real-time performance for texture transfer of video stream. In addition, to enhance the artistic quality, we calculate the level of abstraction using visual saliency and integrate it with the texture transfer algorithm. Thus, our algorithm can stylize video with perceptual enhancements.

Keywords Style transfer · Non-photorealistic rendering · Texture synthesis · Example-based rendering · Stylization

1 Introduction

Nowadays, owing to the vast usage of Internet and portable devices with built-in camcorders, video has become one of the most popular multimedia in the world. A number of people share and watch videos through the streaming services of video-hosting Web services such as YouTube, Vimeo, Facebook, Twitter. Postprocessing applications that give video frames special visual effects, such as perceptual enhancement and artistic retouching, are not very common in most of the services in spite of its broad use because most of the video-processing techniques are computationally expensive.

Artistic postprocessing of video content is very useful to reproduce video content for specific purposes such as education and entertainment. For instance, artistic filters are suitable for educational videos in order to arouse the audience's interest. If such filters are offered on-the-fly while video frames are being played, the audience can enjoy the content with various styles regardless of the original style of the video. In this case, the filters will be useful not only for audience but also for the video content creator, as they can access the result frames with such artistic effects immediately. Hence, developing a real-time artistic filtering technique for video is required.

Texture transfer technique, which takes two images—source and target—and replaces high-frequency part of the target with the texture of the source, has a powerful application that transfers the artistic style of source to the target while preserving the entire structure of the target. According to the style of the source, artistic images with various styles can be generated without specific algorithms required for each artistic style. Therefore, texture transfer is suitable to be employed for the artistic postprocessing of video contents.

However, the texture transfer algorithm synthesizes each pixel in a scan-line order. Following the scan-line order, previously synthesized pixels affect the next synthesizing steps. Thus, this dependency between pixels is an obstacle to parallelization for a single instruction multiple data machine. Although the texture transfer algorithm is designed for fast computation, its performance is not sufficient for a real-time application. In addition, the temporal

✉ Sanghyun Seo
shseo75@gmail.com

Dongwann Kang
dkang@bournemouth.ac.uk

1 Faculty of Science and Technology, Bournemouth University, Poole BH12 5BB, Dorset, UK

2 Division of Media Software, Sungkyul University, Anyang-si, Kyunggi-do, Korea

coherence, which is an essential requirement in stylized animation, must be maintained to apply artistic effects in video frames. In order to maintain the temporal coherence, estimating motions between adjacent frames is necessary, which is computationally expensive.

In this study, we propose a parallel framework for real-time artistic style transfer of a video stream to address issues mentioned above and achieve high real-time performance. First, to reduce the dependency between synthesizing pixels, which is an obstacle to parallelization, we propose a new kernel for synthesizing texture that has horizontally low dependency. The computational performance is simultaneously enhanced by applying this kernel horizontally. Second, we propose a motion approximation method using downscaled frames to decrease a computation time for estimating motions between video frames. Through these approaches, we achieve high real-time performance while transferring artistic style both spatially and temporally. Lastly, to enhance the artistic effect of our method, we propose a perceptual attention-based method for adjusting the level of abstraction (LoA) in the texture transfer algorithm.

The remainder of this study is organized as follows. In Sect. 2, we provide an overview of studies related to texture transfer and artistic stylization of videos. We then present the details of our parallel framework for texture transfer in Sect. 3. The results of our method and comparison of the performance are given in Sect. 4. Finally, we conclude with a summary of our ideas and discuss future work in Sect. 5.

2 Related work

In computer graphics, the texture synthesis technique has been widely studied [6, 8, 9, 17, 18]. This technique generates random high-quality texture from small source texture. However, it focuses on constructing large texture using self-similarity rather than artistic stylization, which is the main focus of this study.

Image analogy [10] is the first approach to stylize an input image using the texture of an example image. This approach synthesizes spatially coherent texture using multiscale autoregression. Although this approach effectively generates artistic results, the computational performance is significantly low. Hence, it is not suitable for real-time applications. Moreover, this algorithm requires a pair of example images: One image is the filtered version of the other. Therefore, the usability is limited.

Ashikhmin [3] proposed a fast texture transfer method, which is an extended version of the local texture synthesis technique [2]. His method effectively transfers the artistic style of the source to the target, requiring only one source image instead of a pair of source images. Although the computational performance of his method is greatly enhanced compared with other methods, it does not attain real-time performance. In this study, based on the method, we propose a parallel framework for a real-time application.

Lee et al. [16] proposed an extended version of Ashikhmin's [3] method. They focused on a directional effect of artistic texture. Their algorithm transfers the texture while preserving the directionality of the target. Ye et al. [25] also presented another extended version of texture transfer, which can be applied to a video input with temporal coherence. In order to enhance temporal coherence, their approach utilizes motions estimated between frames. Along with the motions, the approach utilizes the candidates from synthesized previous frame; thus, temporal coherence can be effectively maintained. Kang et al. [11] extended Ye et al.'s approach to transfer the texture of the source image to the target video while retaining the directionality of the target video. In this study, based on these approaches, which enhance temporal coherence, we aim to enhance real-time computational performance.

As an application of texture transfer, the artistic style transfer is one of the example-based stylization approaches, which generate a result similar to an example image. These approaches include color transfer [4, 21, 23, 24] which is a color correction method for borrowing one images color characteristics from another, photomosaic [13–15, 22] which represents a target image by assembling given photographic examples, collage [12]. In this study, we concentrate mainly on the style transfer approach.

3 Parallel framework for real-time texture transfer of video

3.1 Parallelized texture transfer

The original texture transfer algorithm proposed by Ashikhmin [3] uses two images: target and source. At the initialization step of the algorithm, the initial result is first generated by copying the target. Each pixel in the result is replaced by the pixel randomly selected from the source. At this time, a mapping function $m(x)$ records a mapping from a pixel x in the target to randomly selected pixels in the source. Then, the algorithm considers every pixel of the result in a scan-line order. For each pixel p, the following steps are processed to find the best pixel which satisfies luminance similarity between target and source and texture similarity between source and result being generated.

1. Obtain a set of candidates, $Q\{m(p)+(q-p)|q \in K_L(p)\}$, where $K_L(x)$ denotes the L-shaped kernel shown in Fig. 1b.

2. A randomly selected pixel in source is inserted into Q with a pre-defined probability.
3. From Q, find the best candidate q to have minimum value of

$$\begin{aligned} D(p,q) &= D_B(p,q) + D_L(p,q), \\ D_B(p,q) &= \left|\overline{T(x)} - \overline{S(y)}\right|, x \in K_B(p), y \in K_B(q), \\ D_L(p,q) &= \|H_R(p) - H_S(q)\|, \\ H_R(p) &= \left\{R(x) - \overline{R(x)}\middle| x \in K_L(p)\right\}, \\ H_S(p) &= \left\{S(x) - \overline{S(x)}\middle| x \in K_L(p)\right\}, \end{aligned}$$

where $B(x)$ denotes the box kernel shown in Fig. 1a, $T(x)$, $S(x)$ and $R(x)$ denote intensity of pixel in target, source and result, respectively.
4. Replace $R(p)$ as $S(m(q))$.

These steps can be performed iteratively.

As mentioned in Sect. 1, the original texture transfer algorithm is not suitable to parallelization for enhancing the computational performance because of the dependency between the pixels in L-shaped kernels. This algorithm aims to synthesize texture by comparing the texture of the previously synthesized pixel in the scan-line order and the texture of the candidate pixel in the source, thus to discard the dependency as parallelization does not guarantee appropriate functioning of the algorithm. Therefore, to enhance the computational performance while preserving synthesizing quality, parallelization preserving appropriate dependency is required.

L-shaped kernels can be divided into two parts: the protrusion part on the line which lies on the center of the kernel (the red box in Fig. 1b) and the other part (the blue box in Fig. 1b). Among them, the protrusion part is relatively smaller than the other part. Except for the center of the kernel, we exclude this part from the kernel so that the dependency between pixels on that part and the center is decreased, and thus the parallel computation for each pixel on the same line is available. In this study, we call this kernel as the T-shaped kernel, which is denoted by $K_T(x)$. Our algorithm replaces $K_L(x)$ in the abovementioned four steps with $K_T(x)$ and simultaneously performs these on each pixel on the same line (Fig. 2). This parallel algorithm moves to the next line after completely replacing every pixel on the current line by the best candidate. In other words, in our algorithm, each pixel on the same line is processed simultaneously, and each line is processed sequentially. Through this approach, the computational performance of the texture transfer is dramatically improved.

The texture transfer algorithm can be applied iteratively to enhance the synthesizing quality. As the horizontal dependency in our parallel algorithm is relatively lower than the original approach, horizontal artifact can occur. When we apply our algorithm iteratively, we restrain the

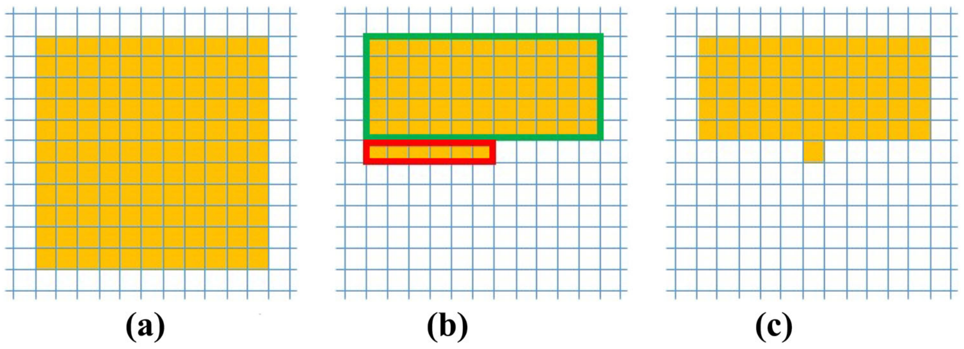


Fig. 1 **a** Box kernel, **b** L-shaped kernel and **c** T-shaped kernel ($r = 6$)

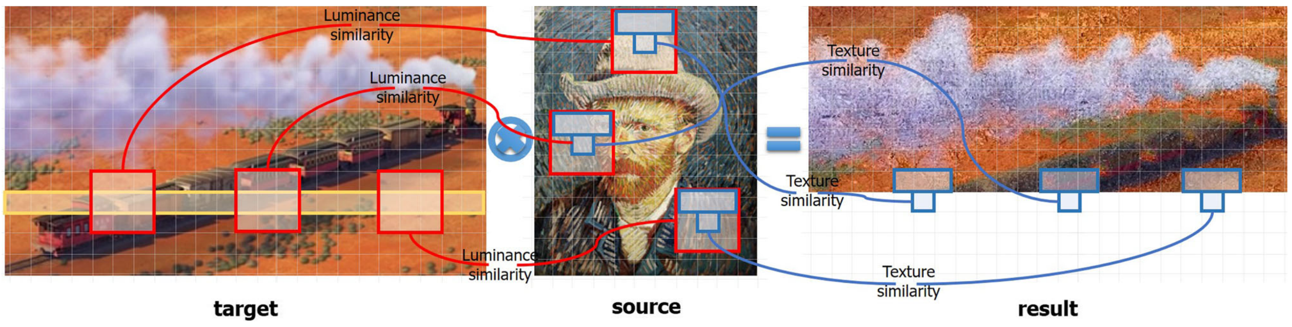


Fig. 2 Parallel framework for real-time texture transfer. For each pixel on the same line (*yellow box*), the texture transfer algorithm with two kernels ($r = 2$) is applied simultaneously, while each line is processed sequentially. Note that pixels are represented as grids on each image

artifact by rotating the target, source and result 90° at the end of each iteration. However, in most of our experiments, reasonable results were obtained without iterations.

3.2 Real-time texture transfer of video stream

In order to apply the texture transfer algorithm to an input video, temporal coherence must be maintained. Performing texture transfer on each video frame independently does not ensure the connectivity between textures on adjacent frames, and thus, significant flickering occurs. Ye et al. [25] proposed a method that brings the pixels that are texture synthesized in the previous frame to the candidates using backward motion and has enhanced temporal coherence. In this study, we employ their approach for our parallel framework to transfer texture to a video stream in real time. First, for the first frame of the video, our algorithm performs single image-based texture transfer presented in Sect. 3.1. Then, from the second frame, steps shown below are performed for each frame.

1. Get $f(x)$, the motion between the previous and current frames.
2. Generate R_i, an initial result of the current frame by replacing $R_i(x)$ with $R_{i-1}(f^{-1}(x))$, and construct $m(x)$, a mapping function to the source.
3. Perform the steps shown below for a single line of the current frame.
   (a) For each pixel on the line, perform the following steps simultaneously.
      (1) Obtain a set of candidates, $Q\{m(p)+(q-p)|q \in K_T(p)\}$ where $K_T(x)$ denotes the T-shaped kernel shown in Fig. 1c.
      (2) A randomly selected pixel in the source is inserted into Q with a pre-defined probability.
      (3) From Q, find the best candidate q to have a minimum value of

$$D(p,q) = D_B(p,q) + D_K(p,q) + D_F(p,q),$$
$$D_B(p,q) = \left|\overline{T(x)} - \overline{S(y)}\right|, x \in K_B(p), y \in K_B(q),$$
$$D_K(p,q) = \|H_R(p) - H_S(q)\|,$$
$$H_R(p) = \left\{R_i(x) - \overline{R_i(x)}\middle| x \in K_K(p)\right\},$$
$$H_S(p) = \left\{S(x) - \overline{S(x)}\middle| x \in K_K(p)\right\},$$
$$D_F(p,q) = C(p) \cdot \left|S(q) - R_{i-1}\left(p + f^{-1}(p)\right)\right|,$$

where $C(x)$ denotes the confidence of the motion quality, which is defined as

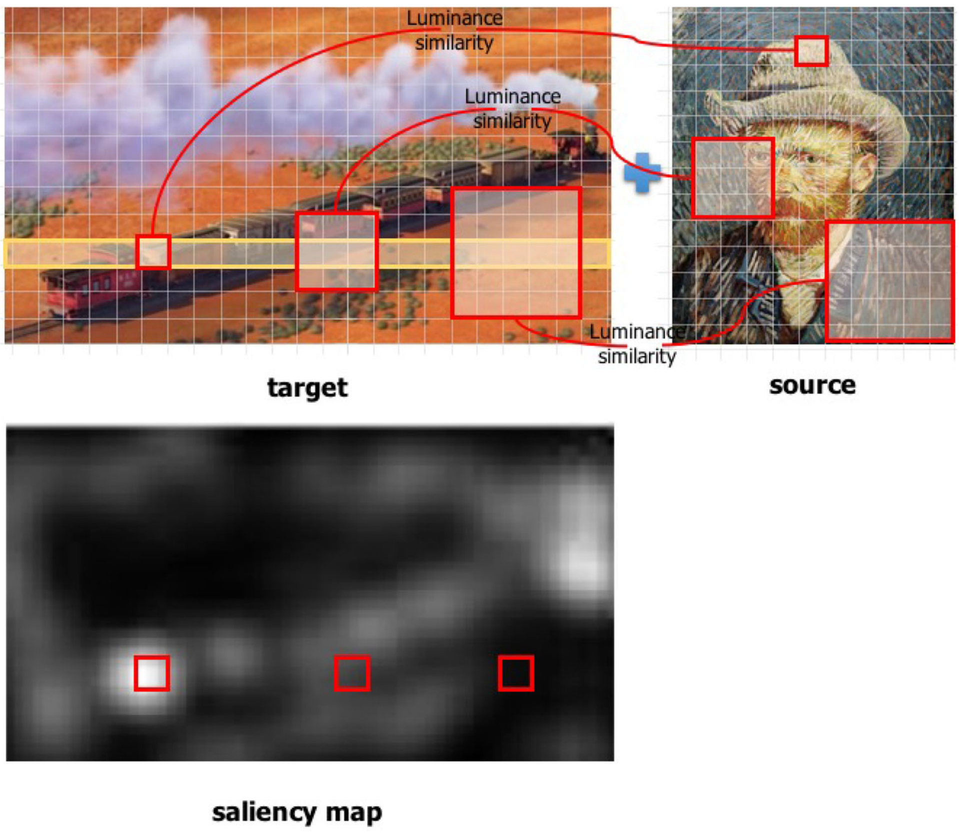


Fig. 3 Texture transfer with adaptive box kernel

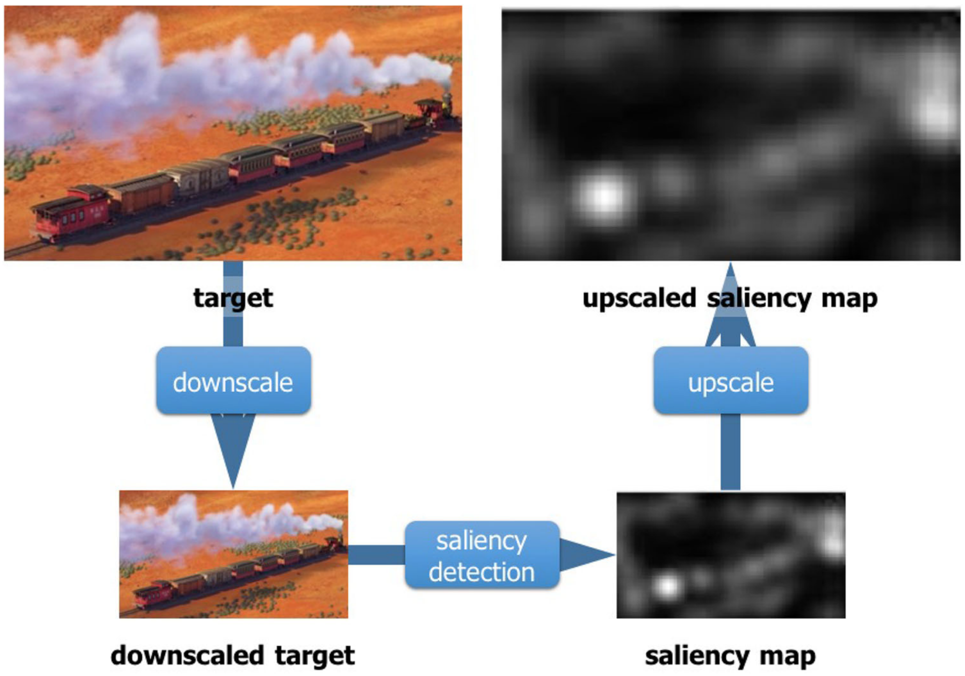


Fig. 4 Fast approximation of visual saliency computation using downscaled saliency map

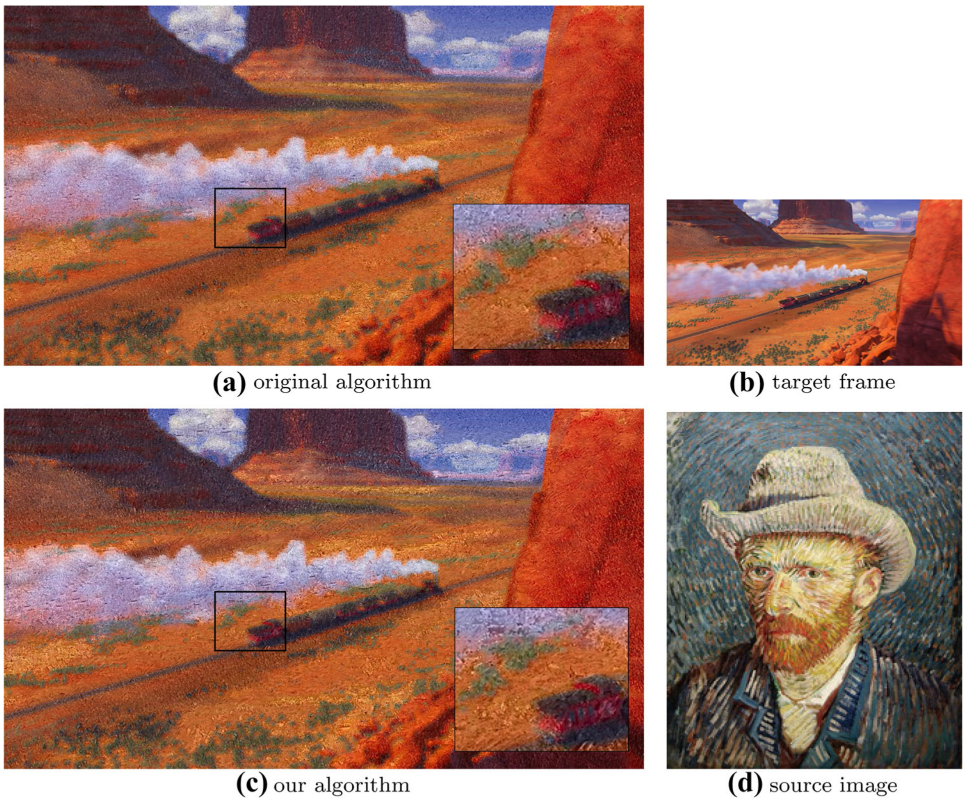


Fig. 5 Results obtained from the original algorithm and our algorithm

$$C(x) = G(\nabla F(x); \sigma_f^2) \cdot G(T_i(x) - T_{i-1}(x + f^{-1}(x)); \sigma_I^2), \quad (4)$$

where $G(\cdot; \sigma^2)$ denotes a zero-mean Gaussian function with variance σ^2.
Replace $R_i(p)$ as $S(m(q))$.

Table 1 Performance of each algorithm according to the resolution of the target and the size of a kernel: default resolution and kernel size are 720p and 5, respectively

		Resolution of target			Kernel size (r)			
		360p	720p	1280p	2	3	5	7
Processing time (s)	Ours	0.01	0.03	0.10	0.01	0.02	0.03	0.04
	Ashikhmin's	0.90	3.21	9.38	1.30	1.91	3.21	4.52

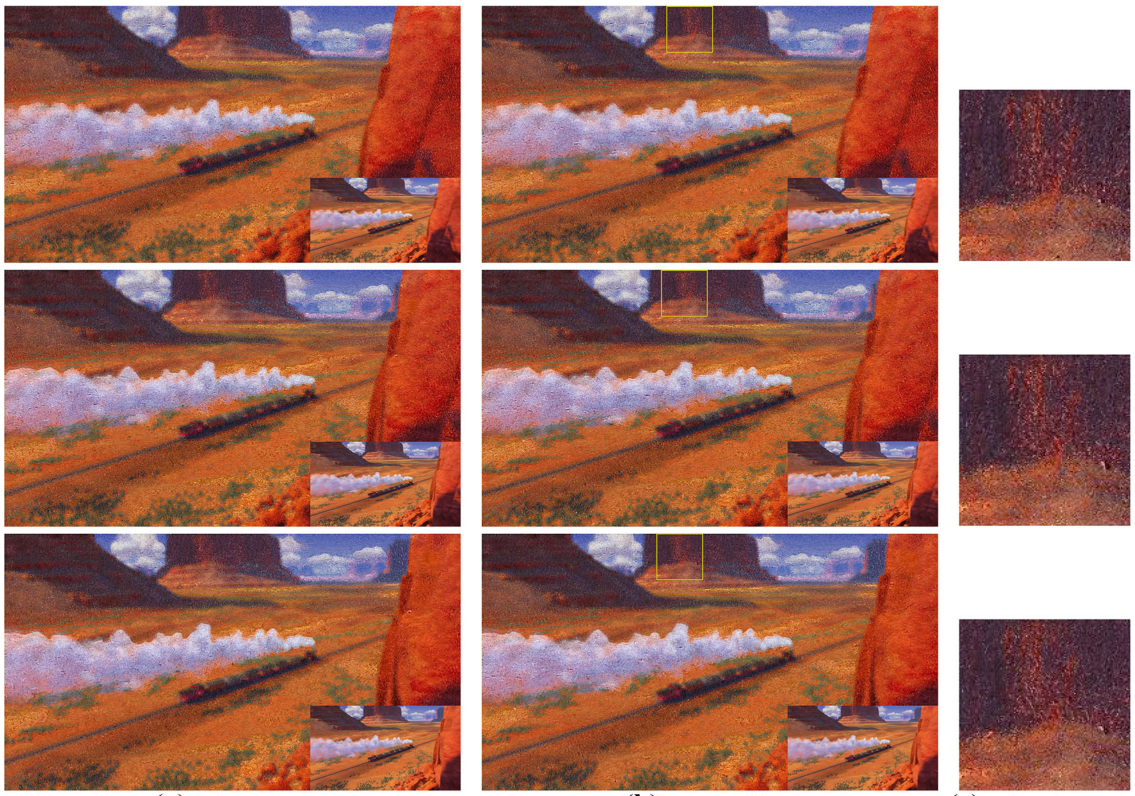

(a) Ye's method **(b)** our algorithm **(c)** yellow boxes in our results

Fig. 6 Result of Ye's method and our algorithm: Each target frame is presented on *lower right corner* of each result. Gogh's "Self-portrait" is used as the source

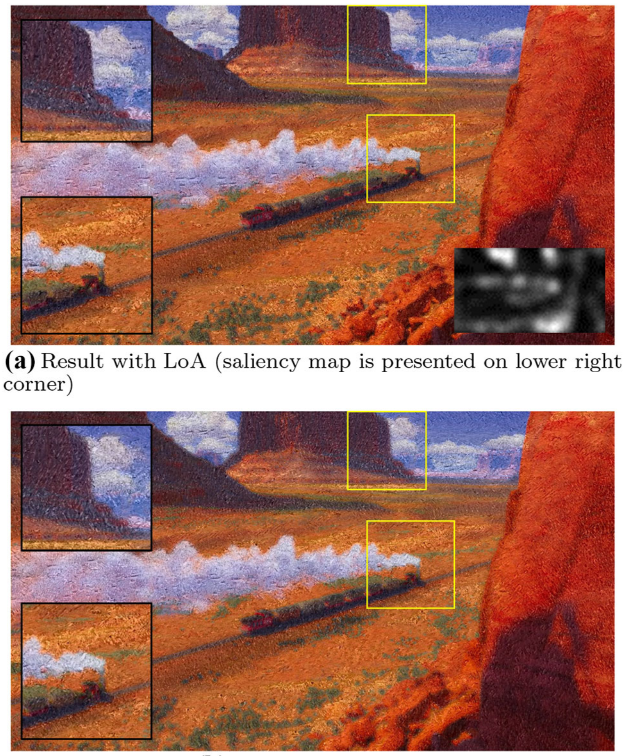
Table 2 Performance of each algorithm according to the resolution of the target and the size of the kernel: default resolution and kernel size are 720p and 5, respectively

		Resolution of target			Kernel size (r)			
		360p	720p	1280p	2	3	5	7
Processing time of each frame (s)	Ours	0.04	0.10	0.46	0.07	0.09	0.10	0.11
	Ye's	16.68	27.73	43.81	25.42	26.89	27.73	28.42

(b) After the processing of every pixel on the line is completed, go to the next single line of current frame and repeat steps (a) and (b).

4. After the textures on every line are transferred, go to the next frame and repeat steps 1–4.

The texture transfer part of our algorithm performs parallel computation so that fast computation can be achieved. However, the motion estimation, which is necessarily required for the video input, has high computation complexity; this contributes to the main bottleneck of our algorithm. In the optical flow method, which is a representative algorithm for estimating motions, a parallel framework for fast computation was already proposed; however, it is not sufficient to attain high real-time performance. In this study, to achieve high real-time performance, we propose a method that reduces the load on the motion estimation by downscaling the resolution of frame. We downscale the resolution of a given input video frame


(a) Result with LoA (saliency map is presented on lower right corner)

(b) Result without LoA

Fig. 7 Comparison of our algorithms with LoA and without LoA: Gogh's "Self-portrait" is used as the source

as $\left(\frac{1}{s_m} \times w, \frac{1}{s_m} \times h\right)$, and estimate the motion using an optical flow algorithm that employs a GPU [19]. Because the resolution of an estimated motion field is smaller than that of original input frames, we upscale the motion field as $(s_m \times w, s_m \times h)$ to be identical to the original input video frame.

3.3 Visual saliency-based level of abstraction

In artistic stylization field, focusing on perceptually meaningful structure in an input image has been one of the significant topics [7]. Because texture transfer algorithm treats the entire area of an image with same level, it is not possible to concentrate on a specific region selectively. In this study, we propose a perceptually inspired method that controls the LoA according to the visual saliency estimated.

Basically, the texture transfer algorithm uses a box kernel for comparing the luminance similarity between the target and source, and therefore, the synthesized texture does not completely reflect the original luminance of the target. This helps in maintaining the low-frequency part only so that it is beneficial to transfer the artistic style of the source effectively. However, the entire area of the synthesized result has a tendency to lose the details of the target. In order to enhance the details adaptively, we use the box kernel with an adaptive size. First, we calculate the value of visual saliency [1], *sal*, from the target frame. We then scale the range of *sal* to $[0 : r - 1]$, where r denotes the radius of the kernels. Finally, we use a box filter with radius, $r_b = r - sal$, when the texture transform is performed on each pixel (Fig. 3). For T-shaped kernel, we use r as its size constantly.

Because we obtain the saliency per frame, the temporal coherence of the saliency value between the frames is not ensured. In order to solve this problem, we blend the saliency value between the pixels on the current and previous frames, which is tracked by the backward motion. Obtaining the saliency value is also not appropriate for real-time applications because of heavy computation. Similar to motion estimation, we enhance the computational performance using saliency values obtained from downscaled frames (Fig. 4).

4 Experimental results

We implemented our parallel framework in Microsoft Visual C++ using CUDA (version 7.5) [20] and OpenCV (version 3.1.0) [5]. Our program requires modern graphics hardware that supports Shader Model 3.0 or higher. We tested our algorithm on Intel i5 3.1 GHz CPU with 8 GB RAM, NVIDIA GeForce GTX980 GPU. In our experiments, we used 720p target videos and kernels with $r = 5$. Following Ashikhmin's [3] approach, we transferred the luminance of the source only while maintaining the chroma of the target frame for our artistic texture transfer application.

We compared our results with the original algorithm for a single image-based parallel framework [3]. As shown in Fig. 5, noticeable difference between the original algorithm using an L-shaped kernel and our algorithm using a T-shaped kernel cannot be found, even though only one iteration is performed. On the other hand, the computational performance is dramatically enhanced as shown in Table 1. For a single image, our algorithm successfully transfers the artistic style of the source in real time.

We compared our results with Ye's method for a parallel framework with a video stream [25]. In this experiment, we used only one iteration and $s_m = 2$ for motion field. Similar to the previous experiment, our algorithm generated reasonable transferred results with temporal coherence as shown in Fig. 6. However, our parallel framework with fast motion estimation achieved high computational performance as compared with Ye's method (Table 2). In our experimental environment, most of the video streams could be stylized in real time through our algorithm.

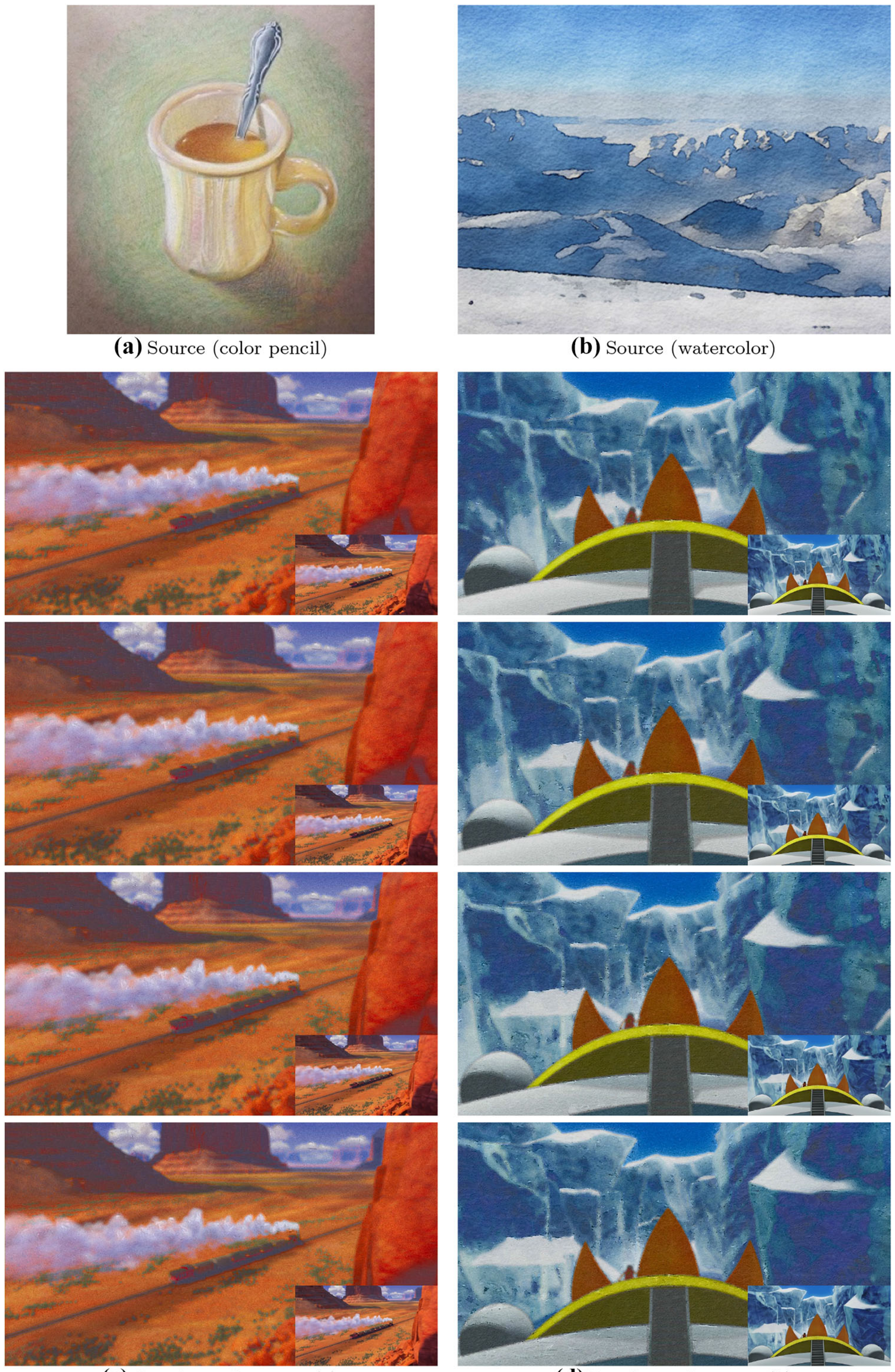

(a) Source (color pencil) (b) Source (watercolor) (c) Results with source (a) (d) Results with source (b)

Fig. 8 Other results using various sources

Figure 7 shows our perceptually inspired results with the adaptive LoA based on visual saliency. Compared with the method that does not use saliency, this approach preserved more details of the target frame at visually significant regions. In this experiment, we used a 1/4 scale for saliency field calculation.

Figure 8 shows other examples.

5 Conclusion

In this study, we presented a real-time texture method for artistic style transfer of video stream. A parallel framework is proposed using a T-shaped kernel to enhance the computational performance. For accelerated motion estimation, which is necessarily required for maintaining the temporal coherence, we presented a method using a downscaled motion field. Through these approaches, we successfully achieved high real-time performance for texture transfer of video stream. In addition, to enhance the artistic quality, we calculated the LoA using visual saliency and integrated it with the texture transfer algorithm; hence, our algorithm can stylize video with perceptual enhancements.

In our future work, we will study more artistic effects that can be integrated with our parallel framework of texture transfer. For instance, the direction of texture [16] is one of the significant artistic factors. We will focus on enhancing the temporal coherence of the texture direction as well as the performance.

Acknowledgments This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1A6A3A03010386) and Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) and Research Development Program 2015.

References

1. Agrawal, R., Gupta, S., Mukherjee, J., Layek, R.K.: A GPU based real-time CUDA implementation for obtaining visual saliency. In: Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing, p. 1. ACM (2014)
2. Ashikhmin, M.: Synthesizing natural textures. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D '01, pp. 217–226. ACM, New York (2001)
3. Ashikhmin, M.: Fast texture transfer. IEEE Comput. Graph. Appl. **23**(4), 38–43 (2003)
4. Bae, S., Paris, S., Durand, F.: Two-scale tone management for photographic look. ACM Trans. Graph. **25**(3), 637–645 (2006)
5. Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. 'O'Reilly Media, Inc., Sebastopol (2008)
6. De Bonet, J.S.: Multiresolution sampling procedure for analysis and synthesis of texture images. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, pp. 361–368. ACM Press/Addison-Wesley Publishing Co, New York, NY (1997)
7. DeCarlo, D., Santella, A.: Stylization and abstraction of photographs. ACM Trans. Graph. **21**(3), 769–776 (2002)
8. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 341–346. ACM (2001)
9. Eisenacher, C., Lefebvre, S., Stamminger, M.: Texture synthesis from photographs. In: Computer Graphics Forum, vol. 27, pp. 419–428. Wiley Online Library (2008)
10. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 327–340. ACM (2001)
11. Kang, D., Kong, P., Yoon, K., Seo, S.: Directional texture transfer for video. Multimed. Tools Appl. **74**(1), 245–258 (2015)
12. Kang, D., Seo, S.: An artistic photographic collage on a mobile device. Multimed. Tools Appl. pp. 1–12 (2014)
13. Kang, D., Seo, S., Ryoo, S.: A parallel framework for fast photomosaics. IEICE Trans. Inf. Syst. **94**(10), 2036–2042 (2011)
14. Kang, D., Seo, S., Ryoo, S., Yoon, K.: A study on stackable mosaic generation for mobile devices. Multimed. Tools Appl. **63**(1), 145–159 (2013)
15. Kim, J., Pellacini, F.: Jigsaw image mosaics. ACM Trans. Graph. **21**(3), 657–664 (2002)
16. Lee, H., Seo, S., Yoon, K.: Directional texture transfer with edge enhancement. Comput. Graph. pp. 81–91 (2011)
17. Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. ACM Trans. Graph. **25**(3), 541–548 (2006)
18. Liang, L., Liu, C., Xu, Y.-Q., Guo, B., Shum, H.-Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20**(3), 127–150 (2001)
19. Marzat, J., Dumortier, Y., Ducrot, A.: Real-time dense and accurate parallel optical flow using CUDA (2009)
20. Nvidia, C.: Compute unified device architecture programming guide (2007)
21. Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Comput. Graph. Appl. **21**(5), 34–41 (2001)
22. Silvers, R., Hawley, M.: Photomosaics. Henry Holt and Co., Inc., New York (1997)
23. Wang, B., Yu, Y., Xu, Y.-Q.: Example-based image color and tone style enhancement. ACM Trans. Graph. **30**(4), 64:1–64:12 (2011)
24. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. ACM Trans. Graph. **21**(3), 277–280 (2002)
25. Ye, N., Sim, T., Miao, X.: Video stylization by single image example. In: Image Processing (ICIP), 2010 17th IEEE International Conference on, pp. 3993–3996. IEEE (2010)

Dongwann Kang received his Ph.D. degree in Chung-Ang University, South Korea, in 2013. He also received his B.S. and M.S. degrees in Computer Science and Engineering from Chung-Ang University in 2006 and 2008. He was the research fellow in Chung-Ang University from March 2013 to June 2015. Now, he is the postdoctoral researcher in the School of Design, Engineering and Computing (DEC), Bournemouth University, UK. His research interests include stylization, emotional computing, image manipulation and GPU processing.

Feng Tian received the Ph.D. degree from Xi'an Jiaotong University, China. Currently, he is an Associate Professor Bournemouth University (BU), UK. He was an Assistant Professor in Nanyang Technological University in Singapore before joining BU in 2009. His current research interests include computer graphics, computer animation, augmented reality, image processing and cloud computing.

Sanghyun Seo received his B.S. degrees in Computer Science and Engineering from Chung-Ang University, Seoul, Republic of Korea, in 1998 and M.S. and Ph.D. degrees in GSAIM Dep. at Chung-Ang University in 2000 and 2010, respectively. He was senior researcher at G-Inno System from 2002 to 2005. He was the postdoctoral researcher at Chung-Ang University in 2010, the postdoctoral researcher at LIRIS Lab, Lyon 1 University, from February 2011 to February 2013, and senior researcher at the ETRI (Electronics and Telecommunications Research Institute), from March 2013 to February 2016. Now, he is an Assistant Professor in Sungkyul University, Korea. His research interests are in the area of computer graphics and non-photorealistic rendering, 3D GIS system and game technology.